01

Testando formulários complexos

Transcrição

Começando deste ponto? Você pode fazer o DOWNLOAD (https://s3.amazonaws.com/caelum-online-public/PM-74/testes-de-sistema-cap3.zip) do projeto completo do capítulo anterior e continuar seus estudos a partir deste capítulo.

Vamos continuar testando nossa aplicação; agora é a vez do cadastro de leilão. Veja que essa tela é bem mais complexa: ela contém combos, checkboxes, e etc. Será que o Selenium consegue lidar com todos esses elementos?

A resposta é sim! Vamos lá! Vamos escrever nosso Page Object que lida com a tela de Novo Leilão. Para adicionarmos um novo leilão, precisamos passar as seguintes informações: nome, valor inicial, usuário e marcar se o objeto é usado. A imagem abaixo mostra a tela de cadastro:

Para preencher uma caixa de texto, já sabemos como fazer:

```
WebElement nome = driver.findElement(By.name("leilao.nome"));
WebElement valor = driver.findElement(By.name("leilao.valorInicial"));

nome.sendKeys("Dono do Produto");
valor.sendKeys("123,45");
```

Já o usuário é um combobox. Para selecionarmos uma opção no combo, devemos fazer uso da classe `Select`. Com esse objeto em mãos, podemos pedir para que ele selecione uma determinada opção. Por exemplo, se quisermos selecionar o usuário "João" (nome que aparece no combo), fazemos:

```
Select usuario = new Select(driver.findElement(By.name("leilao.usuario.id")));
usuario.selectByVisibleText("João");
```

Com o checkbox, é mais fácil. Basta clicarmos nele:

```
WebElement usado = driver.findElement(By.name("leilao.usado");
usado.click();
```

Pronto! Agora sabemos como preencher nosso formulário por inteiro. Vamos escrever nosso Page Object:

```
public class NovoLeilaoPage {

    private WebDriver driver;

    public NovoLeilaoPage(WebDriver driver) {
        this.driver = driver;
    }

    public void preenche(String nome, double valor, String usuario, boolean usado) {

        WebElement txtNome = driver.findElement(By.name("leilao.nome"));
        WebElement txtValor = driver.findElement(By.name("leilao.valorInicial"));

        txtNome.sendKeys(nome);
        txtValor.sendKeys(String.valueOf(valor));

        WebElement combo = driver.findElement(By.name("leilao.usuario.id"));
        Select cbUsuario = new Select(combo);
        cbUsuario.selectByVisibleText(usuario);

        if(usado) {
            WebElement ckUsado = driver.findElement(By.name("leilao.usado"));
            ckUsado.click();
        }

        txtNome.submit();
    }

}
```

Excelente. Podemos usar a mesma estratégia do teste anterior: criar um PageObject para representar a listagem de leilões e chegar na tela de Novo Leilão por ela. Vamos também já colocar um método `existe()` para nos dizer se existe um produto nessa listagem:

```
class LeiloesPage {

    private WebDriver driver;

    public LeiloesPage(WebDriver driver) {
        this.driver = driver;
    }

    public void visita() {
        driver.get("http://localhost:8080/leiloes");
    }

    public NovoLeilaoPage novo() {
        // clica no link de novo leilao
        driver.findElement(By.linkText("Novo Leilão")).click();
        // retorna a classe que representa a nova pagina
        return new NovoLeilaoPage(driver);
    }

    public boolean existe(String produto, double valor, String usuario,
            boolean usado) {
```

```
        return driver.getPageSource().contains(produto) &&
                driver.getPageSource().contains(String.valueOf(valor)) &&
                driver.getPageSource().contains(usado ? "Sim" : "Não");

    }
}
```

Vamos agora ao nosso teste. Devemos cadastrar um leilão:

```
public class LeiloesSystemTest {

    private WebDriver driver;
    private LeiloesPage leiloes;

    @Before
    public void inicializa() {
        this.driver = new FirefoxDriver();
        leiloes = new LeiloesPage(driver);
    }

    @Test
    public void deveCadastrarUmLeilao() {

        leiloes.visita();
        NovoLeilaoPage novoLeilao = leiloes.novo();
        novoLeilao.preenche("Geladeira", 123, "Paulo Henrique", true);

        assertTrue(leiloes.existe("Geladeira", 123, "Paulo Henrique", true));

    }
}
```

Mas o problema é: da onde virá esse usuário? Precisamos cadastrar o usuário "Paulo Henrique" antes de cadastrarmos um leilão para ele. Como faremos isso? Será uma boa ideia já termos alguns dados populados, prontos para os testes?

Uma boa prática de testes é fazer com que a bateria de testes seja inteiramente responsável por montar o cenário que precisa para o teste. Dessa forma, cada teste sabe qual cenário precisa montar. Ou seja, vamos fazer com que um usuário seja adicionado antes de adicionarmos um novo leilão. Para isso, usaremos o `LeiloesPage`:

```
public class LeiloesSystemTest {

    private WebDriver driver;
    private LeiloesPage leiloes;

    @Before
    public void inicializa() {
        this.driver = new FirefoxDriver();
        leiloes = new LeiloesPage(driver);

        UsuariosPage usuarios = new UsuariosPage(driver);
        usuarios.visita();
        usuarios.novo().cadastra("Paulo Henrique", "paulo@henrique.com");
    }
```

```
    @Test
    public void deveCadastrarUmLeilao() {

        leiloes.visita();
        NovoLeilaoPage novoLeilao = leiloes.novo();
        novoLeilao.preenche("Geladeira", 123, "Paulo Henrique", true);

        assertTrue(leiloes.existe("Geladeira", 123, "Paulo Henrique", true));

    }
}
```

Pronto! O teste passa! Veja como é fácil testar formulários complexos, com diferentes tipos de entradas de dados. Além disso, lembre-se sempre de fazer com que seus testes sejam independentes, ou seja, eles devem ser responsáveis por todo o processo, desde a criação do cenário até a validação da saída.